Flex Serv Manuf J (2015) 27:585–605
DOI 10.1007/s10696-015-9210-x

An estimation of distribution algorithm and new computational results for the stochastic resource-constrained project scheduling problem

Chen Fang · Rainer Kolisch · Ling Wang · Chundi Mu

Published online: 13 January 2015

Abstract In this paper we propose an estimation of distribution algorithm (EDA) to solve the stochastic resource-constrained project scheduling problem. The algorithm employs a novel probability model as well as a permutation-based local search. In a comprehensive computational study, we scrutinize the performance of EDA on a set of widely used benchmark instances. Thereby, we analyze the impact of different problem parameters as well as the variance of activity durations. By benchmarking EDA with state-of-the-art algorithms, we can show that its performance compares very favorably to the latter, with a clear dominance in instances with medium to high variance of activity duration.

Keywords Stochastic resource-constrained project scheduling · Estimation of distribution algorithm · Permutation-based local search · Impact of problem parameters

1 Introduction

The stochastic resource-constrained project scheduling problem (SRCPSP) is concerned with scheduling a set of precedence related activities with stochastic durations subject to scarce resources, such that the expected duration of the project (makespan) is minimized (see, e.g., Herroelen and Leus 2005). In case of deterministic activity durations the SRCPSP reduces to the RCPSP. The latter has

C. Fang · L. Wang · C. Mu
Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University, Beijing 100084, China

R. Kolisch (✉)
TUM School of Management, Technische Universität München, Arcisstr. 21, 80333 Munich, Germany
e-mail: rainer.kolisch@tum.de

been shown by Blazewicz et al. (1983) to be NP-hard. Research on the RCPSP has long been employed and its outcome is comprehensive (see for example Brucker et al. 1999; Neumann et al. 2003; Kolisch and Hartmann 2006 as well as Hartmann and Briskorn 2010). However, solving the RCPSP is of limited use for the SRCPSP due to the following reasons. Firstly, a schedule which provides a start time for each activity cannot be employed in the case of stochastic activity durations. Secondly, Heller (1981) shows that project duration is systematically underestimated if the RCPSP is solved for expected activity durations instead of obtaining the expected project duration, subject to the distribution of the activity durations, i.e. $C_{\max}(E(d_1),\ldots,E(d_J)) \leq E(C_{\max}(d_1,\ldots,d_J))$. The difference between $C_{\max}(E(d_1),\ldots,E(d_J))$ and $E(C_{\max}(d_1,\ldots,d_J))$ may become arbitrarily large with increasing number J of activities and increasing variances of activity durations. Thirdly, project planners are not only interested in the expected project makespan but in more informative measures such as the entire distribution of the project makespan or value at risk measures. Hence, there is a legitimate interest in conducting research on the SRCPSP. Nevertheless, research on this topic is still limited.

For solving the SRCPSP, one has to resort to so-called scheduling policies (see Möhring et al. 1984) to which we will also refer simply as policies in the following. The latter transforms a scenario, which for each activity yields a duration drawn from the distribution function of that activity into a schedule, i.e. a start time for each activity. Stork (2000) provides a survey of different classes of policies including information on dominance relationships. For heuristics, two policy classes—neither of which predominates—are most common: Activity-based policies and resource-based policies. Stork (2001) has shown that when interpreted as a function which maps a vector of activity durations into a vector of activity start times resource-based policies are neither monotone nor continuous. As a consequence, the majority of researchers have refrained from employing resource-based policies.

Tsai and Gemmill (1998) proposed a tabu search algorithm for solving the RCPSP and the SRCPSP using resource-based policies, although the latter are not explicitly mentioned by the authors. Stork (2000) developed an exact branch-and-bound algorithm for solving the SRCPSP employing three different classes of policies, amongst them activity-based policies. Ballestin (2007) proposed a genetic algorithm (GA) for the SRCPSP which employs activity-based policies. His computational results show that there is a considerable gap between the deterministic makespan $C_{\max}(E(d_1),\ldots,E(d_J))$ and the expected makespan $E(C_{\max}(d_1,\ldots,d_J))$ for the RCPSP which is increasing with increasing variance of the activity durations. These findings are in line with the results of Heller (1981). Ballestin and Leus (2009) developed a greedy randomized adaptive search procedure (GRASP) which employs activity-based policies. They examined various objective functions related to timely project completion as well as their correlation. They also studied the distribution of the makespan as obtained by GRASP showing that in most cases, all activity distributions, except the exponential distribution, lead to a normal distributed project makespan. Ashtiani et al. (2011) propose a new class of policies, so-called pre-processor policies, a variant of the class of resource-based

policies. They employed pre-processor policies within a genetic algorithm. Pre-processor policies make a number of prior sequencing decisions and thus add some extra precedence relations to the existing ones in order to resolve potential resource conflicts. The remaining resource conflicts are dynamically resolved by employing a resource-based policy.

The estimation of distribution algorithm (EDA) is a newly proposed algorithm framework for stochastic optimization (see Larranaga and Lozano 2002). As such it is in general suited for solving the SRCPSP. Unlike GA, which explicitly applies genetic operators such as crossover and mutation to produce a new generation, EDA reproduces a new generation implicitly by sampling from a probability distribution, which captures features of good solutions. The best solutions derived are then taken in order to further improve the probability distribution towards its ability to sample superior solutions. Thus far, EDA has been applied to a variety of optimization problems, amongst others flow-shop scheduling (see Jarboui et al. 2009), multi-mode resource-constrained project scheduling (see Wang and Fang 2012) and rostering (see Aickelin et al. 2007). EDA shares some common characteristics with the cross-entropy method (see Rubinstein and Kroese 2004), which has been used by Bendavid and Golany (2009) to solve a stochastic project scheduling problem different from the SRCPSP. In this paper we propose EDA to solve the SRCPSP employing the resource-based scheduling policy. The choice of the latter has been inspired by the promising results recently obtained by Ashtiani et al. (2011). Our algorithm is characterized by a novel way of defining and updating the probability distribution. Furthermore, a permutation-based local search method is applied to the best individuals to exploit their neighborhood. Computational results and comparisons with existing algorithms demonstrate the effectiveness of our EDA approach.

The remainder of the paper is organized as follows: In Sect. 2, the SRCPSP is described, and in Sect. 3, the proposed EDA is introduced. Computational results and comparisons are provided in Sect. 4. Finally, we end the paper with some conclusions in Sect. 5.

2 Definitions and problem statement

2.1 The stochastic RCPSP

The (deterministic) RCPSP is concerned with the scheduling of project activities to minimize the project's makespan. The RCPSP can be stated as follows. A project consists of J activities labeled $j = 1, \ldots, J$, where the deterministic duration of activity j is denoted by dd_j. There exist precedence relationships between some activities of the project. This relationship is given by sets of immediate predecessors P_j indicating that an activity j may not be started before any of its predecessors $i \in P_j$ is completed, which can be denoted as $i \prec j$. The set of renewable resources is referred to as K. For each resource $k \in K$, the per-period-availability R_k is assumed to be constant. Activity j requires r_{jk} units of resource k in each period of its non-preemptable duration. The activities $j = 0$ and $j = J + 1$ are dummy activities,

which represent the start and end of the project, respectively. It is assumed that the dummy activities do not require any resources and their durations are equal to zero. The set of all activities including the dummy activities is denoted as $J^+ = \{0, \ldots, J+1\}$. The most common objective is to minimize the makespan of the project; this objective is also considered in our paper. The RCPSP can also be represented by a directed acyclic graph $G(N, A)$, where the set of nodes N denotes the activities of the project and the set of arcs A contains all the arcs (i,j) where $i,j \in J^+$, $i \prec j$. A solution for the deterministic RCPSP is a schedule $s = (s_0, s_1, \ldots, s_{J+1})$ which is a vector of start times of activities. A schedule is feasible if the precedence constraint and resource constraint are guaranteed.

Heuristics for RCPSP usually encode the schedule with some representation, for example an activity list. A schedule generation scheme (SGS) is then used to transform the representation to a schedule. Two SGS procedures are used in literature: the parallel SGS and the serial SGS (see Kolisch 1996 as well as Kolisch and Hartmann 1999 for details).

In the Stochastic RCPSP (SRCPSP), the duration D_j of each activity $j \in J^+$ is a random value which follows a known probability distribution. The vector $(D_0, D_1, \ldots, D_{J+1})$ is denoted by D. For the activities $j = 0, J+1$, $P[D_j = 0] = 1$; for the activities $j = 1, \ldots, P[D_j < 0] = 0$ (where $P[e]$ represents the probability of event e), we use vector $d = (d_0, d_1, \ldots, d_{J+1})$ to represent one particular scenario (also termed sample or realization).

2.2 Scheduling policies

In the stochastic RCPSP, we can no longer use a vector of start times of activities to represent a solution because we do not know the exact duration of each activity before it has been finished. As a result, a scheduling policy is needed to decide which activity to choose at each decision time. Decision times are $t = 0$ (the start time of the project) and the finishing times of activities. For each decision time t, only information which has become available up to t can be used to make the decision (non-anticipativity constraint, see e.g. Stork 2001). That means we can only know the durations of activities which have been finished up to time t. After all activities are completed, we obtain a scenario d from the random duration vector D. Consequently, every policy π may alternatively be interpreted as a function $\pi : R^+_{J+2} \rightarrow R^+_{J+2}$ that maps a given scenario d to start times of activities (schedules) $s(d, \pi)$ (Stork 2000). For a given sample d and scheduling policy π, $s_{J+1}(d, \pi)$ denotes the makespan of the project. The objective for the SRCPSP is to select a policy π^* which minimizes the expected makespan $E[s_{J+1}(D, \pi^*)]$.

Various classes of policies have been proposed for the SRCPSP. Some well-known classes of policies are earliest-start policies (Π^{ES}), preselective policies (Π^{PS}), linear preselective policies (Π^{LPS}), resource-based policies (Π^{RB}), activity-based policies (Π^{AB}), and pre-processor policies (Π^{PP}), (see Stork 2001; Ashtiani et al. 2011).

The policy classes Π^{ES}, Π^{PS} and Π^{LPS} are based on the so-called minimal forbidden sets (see Stork 2000 for details). Since the number of minimal forbidden

sets grows exponentially with the number of activities, the computational time becomes unacceptable when dealing with practical project scheduling problems. As a result, policies which do not need the calculation of minimal forbidden sets should be adopted when dealing with real size projects. These are resource-based policies, activity-based policies and pre-processor policies. As will be shown, each of these policies operates in a specific way with an activity list. Hence, with π we denote activity list and policy interchangeably.

A resource-based policy $\pi \in \Pi^{RB}$ considers at any decision time t the not yet started activities in the order of activity list π. At t, some activities will have finished, some activities will be active (that is, they have started but are not yet finished), and some activities will not yet have started. The policy selects the first activity on the list which has not started, whose predecessors in the activity network have already been completed, and for which there is enough capacity left in order to be processed at t. This activity is started at t. Further activities are selected to start at t until no activity is available any more. Then, the new decision time t' is increased to the next finish time of one of the activities which have been active or started at $t<t'$. The counterpart of a resource-based policy for the RCPSP is the parallel SGS. Stork (2001) has shown that policies from the class Π^{RB}, when viewed as function which maps a vector of activity durations into a vector of activity start times, are not always monotone or continuous. That is, the makespan may increase although activity durations are decreasing (so-called Graham anomalies, see Graham 1966).

An activity-based policy $\pi \in \Pi^{AB}$ schedules activities as early as possible in the order of an activity list π but adds the side constraint that $s_i(d) \leq s_j(d)$ if $i \prec_\pi j$ for each scenario d. $i \prec_\pi j$ denotes that activity i precedes activity j in activity list π. The side constraint is necessary to prevent activity j from delaying activity i, although there is $i \prec_\pi j$. Ballestin (2007) speaks of activity-based policies as "stochastic serial SGS" which equals the deterministic serial SGS with the additional side constraints. Sprecher (2000) shows that when employing the activity-based policy to the (deterministic) RCPSP there will always be one activity list which leads to the minimum makespan.

Pre-processor policies Π^{PP} (see Ashtiani et al. 2011) combine unconditional sequencing decisions as made by earliest-start policies Π^{ES} with the real-time dispatching features of resource-based policies Π^{RB}. For this, a pre-process is employed to define extra arcs which are added to the original directed acyclic graph $G(N,A)$.

Since the aim of this paper is to develop an algorithm to solve real size projects, we cannot employ policies from the classes Π^{ES}, Π^{PS} and Π^{LPS} which are based on the minimal forbidden sets. Thus we have to resort to one of the policy classes Π^{RB}, Π^{AB} or Π^{PP}. Based on the observation of Ashtiani et al. (2011) and Fliedner (2011), which showed that policies from the class of resource-based policies performed on average better than policies from the class of activity-based policies, we employ the class of resource-based policies.

3 Estimation of distribution algorithm for the SRCPSP

The estimation of distribution algorithm (EDA) is an evolutionary metaheuristic which has its theoretical foundation in probability theory. For foundations of EDA see Larranaga and Lozano (2002). EDA has been used for a variety of different problems (see Larranaga and Lozano 2002). A number of papers employ estimation of distribution algorithms for scheduling problems. Jarboui et al. (2009) and Zhang and Li (2011) consider the permutation flow shop problem with a minimizing total flow time objective, and Aickelin and Li (2007) as well as Aickelin et al. (2007) treat nurse scheduling and nurse rostering, respectively. Pan and Ruiz (2012) consider makespan minimization for the lot-streaming flow shop problem with setup times. Finally, Wang and Fang (2012) address makespan minimization for multi-mode resource-constrained project scheduling.

In contrast to the genetic algorithm (GA), EDA does not directly generate new solutions by crossover of parent solutions and mutations but by sampling from a probability distribution. The latter depicts the features of a selected set of feasible solutions of the problem. The probability distribution is the core of EDA and termed there as a probability model. In case of an optimization problem with unrelated variables, where each variable can take on a value independently of the value of the other variables, the probability distribution can be immediately derived from the solution itself. For example, the probability distribution for an unconstrained binary optimization problem max $f(x)$ subject to $x \in \{1, 0\}$ could be a vector p of the size of x where p_i is the parameter of the Bernoulli distribution of variable x_i. However, when variables are interrelated by constraints, the probability distribution is defined for an encoding of the solution instead of the solution itself. An encoding for the RCPSP can be, for example, an activity list. The probability distribution of a list could then be a $J \times J$ matrix which gives for each element (i, j) the probability that activity i is placed at position j of the list (see Wang and Fang 2012). In this paper we define the probability distribution for a matrix X which states for each pair of activities (i, j) whether activity i is placed before activity j on the list or not. Details of the definition of the probability distribution are provided in Sect. 3.2. Using the probability distribution for matrix X, we can sample activity lists and thus solutions. Now, starting with some initial probability distribution for matrix X, in each generation EDA samples a number of activity lists and selects an elite set of lists with the best objective function value. The lists in this set are further improved by applying a simple local search procedure. The resulting elite set of lists is then employed in order to update the probability distribution of the next generation. The aim is to improve the estimate of the probability distribution over the generations in terms of its ability to breed high quality solutions. We will now provide the details for the procedure in Sects. 3.1–3.7.

3.1 Representation and fitness function

As a generalization of RCPSP, SRCPSP also shares most of the characteristics of RCPSP. Kolisch and Hartmann (1999) concluded from experimental tests that

procedures for solving the RCPSP based on the activity list representation yield good results. Inspired by this observation, we adopted the activity list representation to encode individuals. Let us denote the activity list as $\pi = [a_0, a_1, \ldots, a_{J+1}]$ where a_j is the activity on position j of the list.

As fitness value of an activity list π we use the expected makespan which we approximate by the average makespan of *nscen* scenarios

$$fitness(\pi) = \frac{1}{nscen} \sum_{n=1}^{nscen} s_{J+1}(d^n, \pi) \tag{1}$$

Ballestin (2007) showed that for a fixed number of generated schedules $s(d^n, \pi)$ using less scenarios *nscen* is beneficial because more individuals can be evaluated. Following Ballestin and Leus (2009) we choose $nscen = 10$ to evaluate an activity list during the search procedure.

3.2 Probability model

As stated at the beginning of Sect. 3, we use an activity list solution representation in order to build the probability model. Let X be a $J \times J$ binary random variable matrix which can be defined as

$$X = \begin{pmatrix} X_{11} & \cdots & X_{1J} \\ \vdots & \ddots & \vdots \\ X_{J1} & \cdots & X_{JJ} \end{pmatrix} \tag{2}$$

with

$$X_{ij} = \begin{cases} 1, & \text{if activity } i \text{ is placed before activity } j \text{ in activity list } \pi \\ 0, & \text{else} \end{cases} \tag{3}$$

Accordingly, we can map an activity list π to X with a function $\{0, 1, \ldots, J+1\}^{J+2} \mapsto \{0, 1\}^{J \times J}$:

$$X = f(\pi) \tag{4}$$

Let us illustrate $X = f(\pi)$ by mapping activity list $\pi = [2, 1, 3]$ into matrix $X = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$. Since for a feasible activity list, activity i is placed before activity j or vice versa, $X_{i,j} + X_{j,i} = 1$ holds for any two activities i, j, $i \neq j$. Furthermore, X contains the transitive closure obtained from a graph which depicts the linear order of the activities in π and thus has $\frac{J \cdot (J-1)}{2}$ non-zero entries, whereas π requires $J + 2$ entries only. Since the entries in X are in $\{0, 1\}$, it follows immediately that the sum of all entries in X is $\frac{J \cdot (J-1)}{2}$.

We can now define the probability matrix

$$\phi = \begin{pmatrix} p(X_{11}=1) & \dots & p(X_{13}=1) \\ \vdots & \ddots & \vdots \\ p(X_{J1}=1) & \dots & p(X_{JJ}=1) \end{pmatrix} \tag{5}$$

where element $p(X_{ij}=1) \in [0,1]$ represents the probability that activity i is placed before j in the list. Since the probability matrix will change during the iteration of the EDA we denote with φ_g and with $p_g = (X_{ij})$ the probability matrix and element (i, j) of the matrix in iteration g, respectively.

3.3 Distribution-based offspring sampling

We can now use probability matrix φ in order to sample activity lists and thus solutions. For this we employ a variant of the parallel SGS (see Kolisch and Hartmann 1999). Starting with the partial list $\pi = [a_0 = 0]$ with activity 0 being in the first position a_0, the set of eligible activities E is defined. An activity j is in E if it is not on the list and each of its predecessors in the activity network is on the list. For each activity $j \in E$ the probability of extending the list with j is calculated according to

$$Prob_i = \frac{\sum_{j \in E} p_g(X_{ij}=1)}{\sum_{i \in E} \sum_{j \in E} p_g(X_{ij}=1)} \tag{6}$$

Algorithm DBOS(φ_g)

$a_0 = 0$;
for *j=0* to *J*
  for *i=0* to *J+1*
    if $i \in E$
      Calculate the selecting probability $Prob_i$ of activity *i* according to Eq. (6);
    else
      $Prob_i = 0$;
  end for
  Select an activity *m* according to the selecting probabilities;
  $a_{j+1} = m$;
end for
Return $\pi = [a_0, a_1, \dots, a_{J+1}]$

Fig. 1 Procedure DBOS

and the next activity on the list is randomly selected based on *Prob*. To illustrate the approach let us consider a project with 3 non-dummy activities and single precedence constraint $2 \rightarrow 3$ as well as the probability matrix $\varphi = \begin{pmatrix} 0 & .3 & .7 \\ .7 & 0 & .7 \\ .3 & .3 & 0 \end{pmatrix}$. For determining activity a_1 we have $E = \{1, 2\}$, $Prob_1 = \frac{0.3}{0.3+0.7} = 0.3$ and $Prob_1 = \frac{0.7}{0.3+0.7} = 0.7$. Figure 1 provides the pseudocode of distribution-based offspring sampling (DBOS).

3.4 Local search strategy

In order to improve solutions we employ a permutation-based local search strategy (PBLS) which is described in Fig. 2. It employs the swap operator proposed by Hartmann (1998) by swapping the position of the i-th and the $(i + 1)$th activity in the list with probability *Pper*, if the two activities are not precedence related.

3.5 Updating mechanism

A population-based updating mechanism is proposed to update the probabilistic matrix φ_g. Firstly, in iteration g the population $\Omega_g : \{\pi_g(1), \pi_g(2), \ldots, \pi_g(M)\}$ of size M is generated according to the matrix φ_g. After evaluating the population, $Q < M$ best individuals are selected from Ω_g to form the elite set $\Omega_g^{Elite} : \{\pi_g^E(1), \pi_g^E(2), \ldots, \pi_g^E(Q)\}$. Secondly, the PBLS is employed to improve each individual of the elite set. Then, the elite set is chosen to update φ_g according to equation

```
Algorithm PBLS(π=[a_0, a_1, ..., a_{J+1}])
for (i=1,2,...,J-1)
    Randomly generate value rand where 0<rand<1;
    if (rand<Pper)
        if ( a_i is not the predecessor of a_{i+1} )
            Swap a_i and a_{i+1} ;
            Evaluate the new π ;
            if (fitness value is improved)
                Record current π and fitness value;
            end if
        end if
    end if
end for
return π
```

Fig. 2 Procedure of PBLS

$$\varphi_{g+1} = (1-\beta)\cdot\varphi_g + \beta\cdot\frac{1}{Q}\sum_{q\in\Omega_g^{Elite}} I_g(q) \tag{7}$$

where β is the learning speed and $I_g(q)$ is the frequency matrix of the q^{th} individual of Ω_g^{Elite} in generation g which is defined as

$$I_g(q) = \begin{pmatrix} \delta_{11g}(q) & \cdots & \delta_{1Jg}(q) \\ \vdots & \ddots & \vdots \\ \delta_{J1g}(q) & \cdots & \delta_{JJg}(q) \end{pmatrix} \tag{8}$$

employing the frequency function $\delta_{ijg}(q)$ according to:

$$\delta_{ijg}(q) = \begin{cases} 1, & \text{if activity } i \text{ is placed before activity } j \text{ in the } q^{th} \text{ individual of } \Omega_g^{Elite} \text{ in generation } g; \\ 0, & else. \end{cases} \tag{9}$$

Note that the updating mechanism always maintains the characteristic $p(X_{ij}=1)+p(X_{ji}=1)=1$ of φ. In contrast to X, the number of non-zero entries is not limited to $\frac{J\cdot(J-1)}{2}$ but typically is $J\cdot(J-1)$.

3.6 Initial population

Hartmann (1998, 2002) has shown that it is advantageous to generate an initial population with the regret-based biased random sampling method of Kolisch (1996) using the latest finish time (LFT) priority rule. Following this advice, we generate an initial population Ω_{Init}. For each of the $q = 1, \ldots, M$ activity lists $\pi_{Init}(q)$ we obtain the frequency matrix $I_{Init}(q)$ according to Eqs. (8)–(9). In order to balance quality and diversity for the initial probability matrix φ_0 we blend these frequency matrices with the uniform probability matrix

$$\varphi_{uniform} = \begin{pmatrix} 0 & 0.5 & \cdots & 0.5 \\ 0.5 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0.5 \\ 0.5 & \ldots & 0.5 & 0 \end{pmatrix} \tag{10}$$

by employing Eq. (7) with parameter β

$$\varphi_0 = (1-\beta)\cdot\varphi_{uniform} + \beta\cdot\frac{1}{M}\sum_{q\in\Omega_0} I_{Init}(q) \tag{11}$$

Note that for the probability matrix of each generation the sum of all entries of φ is $\frac{J\cdot(J-1)}{2}$.

3.7 EDA-procedure for the SRCPSP

Employing the building blocks presented above, we can now give the overall description of EDA for solving the SRCPSP in Fig. 3.

4 Computational results

4.1 Problem instances

We coded the proposed algorithm in C++ using Microsoft Visual Studio 2005. All the experiments were performed on an IBM Thinkpad T61 with a Core 2 T7500 2.2 GHz processor. We used the standard RCPSP dataset J120 from the PSPLIB (see Kolisch and Sprecher 1996) for testing. The problem set J120 contains 600 instances with 120 activities each.

We follow Stork (2001), Ballestin and Leus (2009) and Ashtiani et al. (2011) in the choice of the probability distribution types, means, and variances. The deterministic processing times $d^* \in \mathrm{N}^J$ that appear in J120 are taken as the expected values for the stochastic duration. We work with five distributions: two continuous

Algorithm EDA

Generate the initial population $\Omega_{Init} : \{\pi_{Init}(1), \pi_{Init}(2), \ldots, \pi_{Init}(M)\}$

Generate the initial probability matrix φ_0 according to Eq. (11);

$\pi^{best} = \pi_{Init}(1)$;

$g = 0$;

do

  for i to M

    $\boldsymbol{\pi_g(i) = DBOS(\varphi_g)}$;

  end for

  Evaluate the population $\Omega_g : \{\pi_g(1), \pi_g(2), \ldots, \pi_g(M)\}$ according to Eq. (1);

  Select the Q best individuals to form the elite set $\Omega_g^{Elite} : \{\pi_g^E(1), \pi_g^E(2), \ldots, \pi_g^E(Q)\}$;

  for i to Q

    $\pi_g^E(i) = PBLS(\pi_g^E(i))$;

    if $fitness(\pi_g^E(i)) < fitness(\pi^{best})$

      $\pi^{best} = \pi_g^E(i)$;

    end if

  end for

  Generate the probability matrix φ_{g+1} according to Eqs. (7)-(9);

  $g++$;

while (Stopping condition is not met)

return π^{best}

Fig. 3 EDA for the SRCPSP

uniform distributions with intervals $[d_i^* - \sqrt{d_i^*}; d_i^* + \sqrt{d_i^*}]$ and $[0; 2d_i^*]$; one exponential distribution with mean d_i^*; and two beta distributions with variance $d_i^*/3$ and $d_i^{*2}/3$, both with support $[d_i^*/2; 2d_i^*]$. In the following, we will refer to these five distributions as U1, U2, Exp, B1, and B2, respectively. The variance of these distributions is $d_i^*/3$, $d_i^{*2}/3$, d_i^{*2} $d_i^*/3$, and $d_i^{*2}/3$, respectively. That means that U1 and B1 have relatively little variability, U2 and B2 have medium variability, and Exp has large variability. The parameters (α, β) of the beta distribution are $(d_i^*/2 - 1/3, d_i^* - 2/3)$ and $(1/6, 1/3)$ for B1 and B2, respectively.

We evaluate the quality of the algorithm by the average percentage deviation of $E[s_{J+1}(D, \pi)]$ from the critical path length with the deterministic durations d^*. The expected makespan is obtained by mean of a simulation with 1,000 replications. In order to compare different algorithms fairly, computational effort is measured by the number of generated schedules, which is 5,000 and 25,000 (see Kolisch and Hartmann 2006). Solving one scenario with a resource-based policy will be counted as one generated schedule.

4.2 Setting the parameters of the EDA

We used the Taguchi method of design of experiment (see Montgomery 2009) to determine a set of suitable parameters for the EDA. From the J120 dataset we chose 60 instances according to Xp_q.RCP, where $p = 1, 2, \ldots, 60$ and $q = p - \left\lfloor \frac{p-1}{10} \right\rfloor \times 10$. For each of these instances we chose the U2 distribution since it has a medium level of variability.

The EDA contains four key parameters: the population size of each generation (M), the size of the elite set (Q), the PBLS acceptance rate ($Pper$), and the learning speed (β). With the five levels for each parameter given in Table 1 and using an orthogonal array $L_{25}(5^4)$, we obtain 25 parameter combinations. Each of the 60 instances is solved with each parameter combination and a maximum number of 5,000 and 25,000 schedules as stopping condition. The response R for each parameter combination and stopping criterion is the average deviation of the makespan obtained by the thus parameterized EDA from the critical path based lower bound LB of the instance with deterministic activity durations d^*.

$$R = \frac{1}{60} \sum_{i=1}^{60} \frac{(Makespan_i - LB_i)}{LB_i} \tag{12}$$

Table 1 Combinations of parameter values

Parameters	Factor level				
	1	2	3	4	5
M	100	150	200	250	300
Q	1 % M	2 % M	5 % M	8 % M	10 % M
$Pper$	0.1	0.3	0.5	0.7	0.9
β	0.1	0.3	0.5	0.7	0.9

Figures 4 and 5 present the main effect of the parameter variations on the solution quality. It can be seen that the main effect is rather moderate. Based on the results we set the parameter to $M = 150$, $Q = 1$ % M, $Pper = 0.5$ and $\beta = 0.3$ for the following experiments.

4.3 The impact of the project characteristics and distribution types

In this section, we analyze the impact of project characteristics on the performance of the proposed EDA. According to Kolisch and Hartmann (1999), a full factorial design of the variable parameters including network complexity (*NC*) (3 levels), resource factor (*RF*) (4 levels), and resource strength (*RS*) (5 levels) with 10 replications per cell is adopted to generate a total of $3 \times 4 \times 5 \times 10 = 600$ benchmark problems for J120. *NC* is the average number of non-redundant arcs per node, including the dummy start and finish activity. $RF \in [0, 1]$ gives the average percentage of resources requested per activity, while $RS \in [0, 1]$ measures the strength of the resource constraints, where low values indicate that resource constraints are tight. Besides the distributions U1, U2, Exp, B1, and B2 we also adopt the deterministic case (Deter) for comparison. Only 1 scenario ($nscen = 1$) is needed for calculating the fitness value of an activity list for the deterministic case, whereas 10 scenarios ($nscen = 10$) are used for calculating each fitness value for U1, U2, Exp, B1, and B, respectively. For a fair comparison, 500 schedules and 2,500 schedules are adopted as the stopping conditions for Deter. Tables 2 and 3 as well as Fig. 6 provide information on the average percentage deviation of EDA from the deterministic critical path based lower bound with deterministic durations with respect to the levels of problem parameters and the distribution functions. There are a number of observations which can be made. Firstly, the average

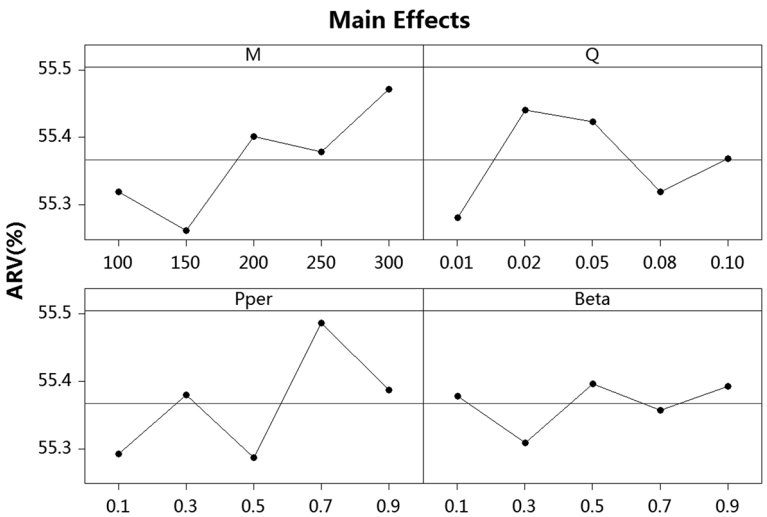

Fig. 4 Factor level trend with 5,000 schedules

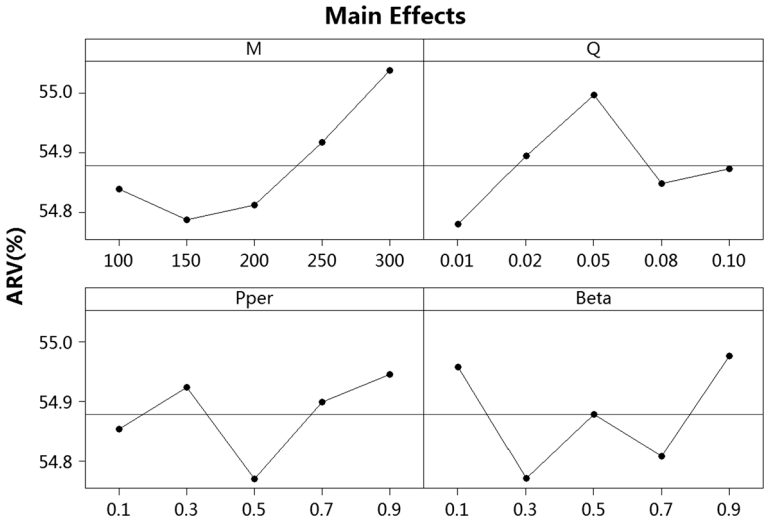

Fig. 5 Factor level trend with 25,000 schedules

Table 2 Average percentage deviation for different project parameters and distributions (5,000 schedules)

Project parameters	U1	U2	Exp	B1	B2	Deter
NC						
1.5	45.65	54.53	69.95	45.98	56.36	39.34
1.8	45.35	54.49	70.45	45.79	56.35	38.83
2.1	50.87	60.61	77.11	51.17	62.16	44.51
RF						
0.25	19.58	31.04	50.73	19.9	32.48	12.31
0.5	44.52	54.15	70.83	44.74	55.5	36.29
0.75	58.3	66.7	80.91	58.74	68.32	52.31
1	66.76	74.3	87.54	67.21	76.87	62.67
RS						
0.1	120.51	131.51	148.07	120.84	133.67	111.33
0.2	60.63	69.3	83.67	60.89	71.27	53.68
0.3	29.31	37.48	52.06	29.69	39.13	23.71
0.4	17.16	26.38	43.17	17.66	27.79	11.87
0.5	8.84	18.04	35.55	9.15	19.59	3.88

deviation from the lower bound (Ave.LB.Dev) increases with increasing variance of the distribution function. This effect is consistent for all problem parameters and parameter levels and confirms the results of Heller (1981) and Ballestin (2007). Secondly, the type of distribution function does not have an impact but the variance of the distribution does. This can be observed for the uniform distribution U1 and

Table 3 Average percentage deviation for different project parameters and distributions (25,000 schedules)

Project parameters	U1	U2	Exp	B1	B2	Deter
NC						
1.5	45.03	54.16	69.62	45.57	55.93	38.28
1.8	44.79	54.07	70.02	45.14	55.88	37.84
2.1	50.16	59.97	76.52	50.42	61.66	42.82
RF						
0.25	19.41	30.89	50.6	19.68	32.34	12.09
0.5	43.89	53.85	70.33	44.09	55.12	34.87
0.75	57.41	65.96	80.36	57.91	67.75	50.75
1	65.94	73.57	86.93	66.48	76.09	60.88
RS						
0.1	119.05	130.47	147.04	119.58	132.74	108.77
0.2	59.95	68.73	83.13	60.2	70.66	52.24
0.3	28.79	37.07	51.73	29.07	38.71	22.62
0.4	16.82	26.12	42.92	17.32	27.54	11.06
0.5	8.7	17.95	35.45	9.02	19.46	3.54

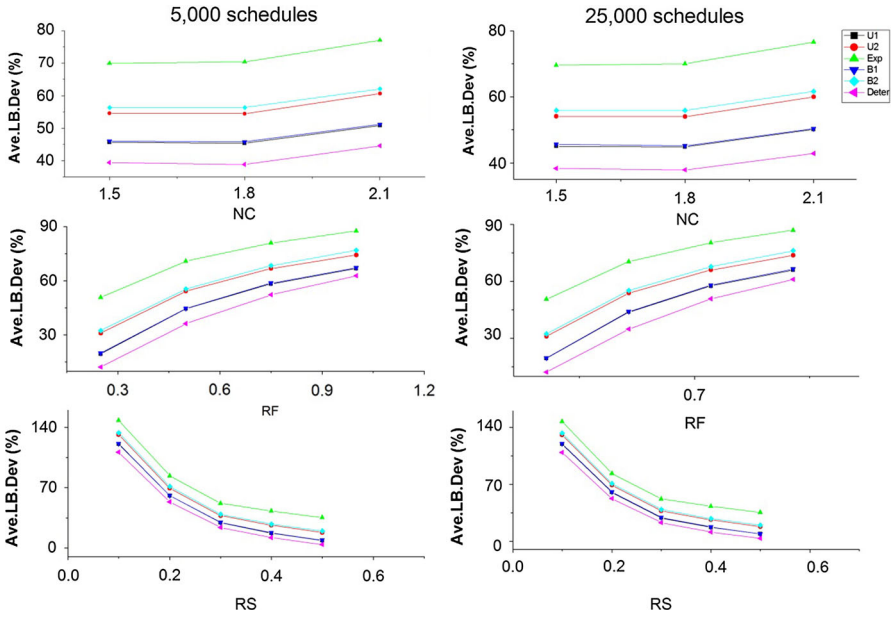

Fig. 6 Average percentage deviation for different project parameters and distributions

the beta distribution B1 which have the same variance of $d_i^*/3$. Thirdly, the average deviation from the lower bound increases for increasing resource factor *RF*, i.e. more resources are requested by an activity, and decreasing resource strength *RS*, i.e. scarcer resources. These findings are basically in line with observations on the impact of the problem parameters on the RCPSP (see Kolisch 1995). The impact of the network complexity *NC* is not as clear. While there is no impact when increasing *NC* from 1.5 to 1.8, there is a decrease of the average deviation from the lower bound when *NC* is increased from 1.8 to 2.1. This effect deviates from previous findings for the deterministic case where Kolisch (1995) did not observe a significant impact from the *NC* on the performance of priority rule based heuristics. However, Kolisch (1995) measured the deviation from the optimal solution while, due to the size of the problems, we are measuring the deviation from the critical path based lower bound. In order to analyze the impact of the lower bound in more detail, Table 4 provides the critical path based lower bound (LB) and the upper bound (UB) generated by the proposed EDA for different project parameters for the deterministic case. Based on the results listed in Table 4, the impacts of different project parameters are illustrated in Figs. 7, 8 and 9. Figure 7 shows that the *NC* has an impact on both the LB and the UB. When *NC* increases, LB and UB increase as well. The reason is that additional (non-redundant) precedence relations lead to an increase in the length of the critical path of the project (LB) as well as a resource-feasible project makespan (UB). Interestingly, the slope of LB decreases for increasing *NC*, while the slope of UB increases. That is, from a medium level of *NC* on, the impact of additional precedence relations on the critical path decreases noticeably, while the impact on the resource feasible makespan increases mildly.

Table 4 LB and UB for different project parameters (deterministic case)

Project parameters	LB	UB (500 schedules)	UB (2,500 schedules)
NC			
1.5	86.78	120.24	119.33
1.8	95.92	132.83	131.88
2.1	102.16	146.78	145.08
RF			
0.25	94.69	105.99	105.8
0.5	95.29	129.41	128.06
0.75	95.51	144.75	143.28
1	94.31	152.97	151.23
RS			
0.1	95.06	199.26	196.8
0.2	93.86	143.99	142.63
0.3	97.62	120.52	119.46
0.4	93.28	104.13	103.39
0.5	94.93	98.5	98.19

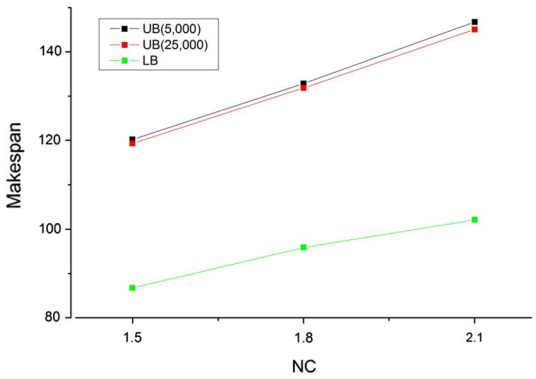

Fig. 7 The impact of NC on LB and UB

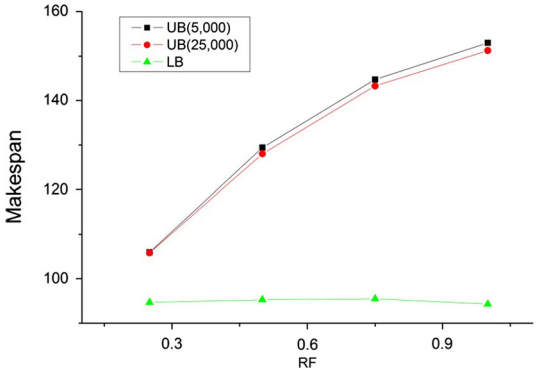

Fig. 8 The impact of RF on LB and UB

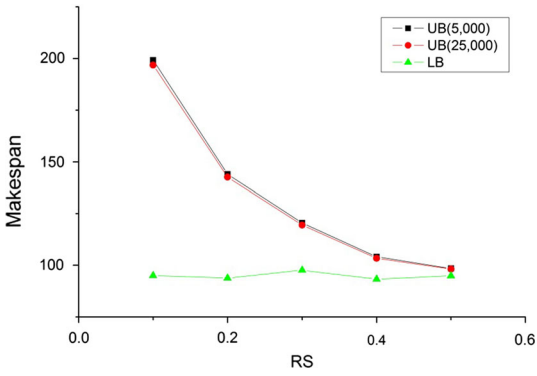

Fig. 9 The impact of RS on LB and UB

This gives evidence for the assumption that the increase in Ave.LB.Dev when increasing *NC* from 1.8 to 2.1 does not primarily stem from a poor performance of EDA, but from the decreasing impact on the critical path based lower bound. Figures 8 and 9 show that the effect of the two resource parameters *RF* and *RS* on

the critical path based lower bound is null, which is as expected. The impact of *RF* and *RS* on the upper bound is considerable.

4.4 Performance comparison with other heuristics

In this section, we compare the EDA with the state-of-the-art algorithms for the SRCPSP. The algorithms compared include the genetic algorithm of Ballestin (2007), denoted as ABGA, the greedy randomized adaptive search procedure proposed by Ballestin and Leus (2009), denoted as ABGR, and the two-phase genetic algorithm of Ashtiani et al. (2011), denoted as PPGA. The comparison results are depicted in Tables 5 and 6 where the best performance for each distribution is set in bold.

Tables 5 and 6 reveal that the EDA outperforms the ABGA and PPGA in all cases. Like the PPGA, the EDA outperforms the ABGR in the medium and high-variability cases (U2, Exp and B2). However, the ABGR is slightly better than the EDA for low variability (U1 and B1). Since ABGR employs an activity-based policy and EDA employs a resource-based policy this might be an indication for the superiority of activity-based policies for problems with small variability of activity durations. This conjecture is backed up by theoretical and experimental results for the deterministic case, i.e. the extreme case of the SRCPSP where variability converges to zero. There, Sprecher (2000) has proven that when scheduling activity lists according to the serial scheduling scheme with side constraints $s_i(d) \leq s_j(d)$ if $i \prec_\pi j$, which is the deterministic counterpart of the activity-based scheduling policy, there is always one activity list with minimum makespan. Kolisch (1996) has proven that the parallel schedule generation scheme, the deterministic counterpart of the resource-based policy, searches in the set of nondelay schedules which does not

Table 5 Comparison with other algorithms (5,000 schedules)

Procedure	Distribution				
	U1	U2	Exp	B1	B2
ABGA (Ballestin 2007)	51.49	78.65	120.22	–	–
ABGR (Ballestin and Leus 2009)	**46.84**	72.58	114.42	**47.17**	75.97
PPGA (Ashtiani et al. 2011)	48.86	58.91	76.03	49.01	58.82
EDA	47.29	**56.54**	**72.50**	47.65	**58.29**

Table 6 Comparison with other algorithms (25,000 schedules)

Procedure	Distribution				
	U1	U2	Exp	B1	B2
ABGA (Ballestin 2007)	49.63	75.38	116.83	–	–
ABGR (Ballestin and Leus 2009)	**45.21**	70.95	112.37	**45.60**	74.17
PPGA (Ashtiani et al. 2011)	47.21	58.07	74.56	47.25	57.95
EDA	46.66	**56.07**	**72.05**	47.04	**57.82**

always include an optimal solution. Kolisch and Hartmann (2006) have experimentally shown that heuristics which employ the serial schedule generation scheme perform better than heuristics which use the the parallel schedule generation scheme, if the runtime and thus the number of generated schedules is sufficiently large.

5 Conclusion and future work

In this paper, we proposed the estimation of distribution algorithm (EDA) for the stochastic resource-constrained project scheduling problem (SRCPSP). The EDA utilizes the statistic information obtained from the elite individuals of the former generation to predict the promising area in the searching space. By adopting a novel probability model and an updating mechanism, the promising area could be tracked effectively. By adopting the permutation-based local search strategy (PBLS), the exploitation ability is further enhanced. Using an experimental design with orthogonal array, suitable parameter settings for the EDA were determined. Simulation results based on the PSPLIB benchmarks and comparisons with some existing algorithms demonstrated the effectiveness of the proposed EDA and the impact of problem parameters and the activity distributions on its performance. Comparing EDA to state-of-the-art heuristics for the SRCPSP, we could show that the proposed procedure is quite competitive and, indeed, yields the best performance if the variance of the activity duration is medium to large. Possible future work is to develop an adaptive EDA with a parameter learning mechanism and to develop a new class scheduling policy for the SRCPSP.

Acknowledgments This paper was written during Chen Fang's one year research stay at the TUM School of Management. The authors thank two anonymous reviewers for their valuable comments. This research has been partially supported by National Key Basic Research and Development Program of China (Grant No. 2013CB329503), National Science Foundation of China (Grant No. 61174189), and Doctoral Program Foundation of Institutions of Higher Education of China (Grant No. 20130002110057).

References

Aickelin U, Li J (2007) An estimation of distribution algorithm for nurse scheduling. Ann Oper Res 155(1):289–309

Aickelin U, Burke EK, Li J (2007) An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering. J Oper Res Soc 58(12):1574–1585

Ashtiani B, Leus R, Aryanezhad M (2011) New competitive results for the stochastic resource-constrained project scheduling problem: exploring the benefits of pre-processing. J Sched 14(2):157–171

Ballestin F (2007) When it is worthwhile to work with the stochastic RCPSP? J Sched 10(3):153–166

Ballestin F, Leus R (2009) Resource-constrained project scheduling for timely project completion with stochastic activity durations. Prod Oper Manag 18(4):459–474

Bendavid I, Golany B (2009) Setting gates for activities in the stochastic project scheduling problem through the cross entropy methodology. Ann Oper Res 172(1):259–276

Blazewicz J, Lenstra JK, Rinnooy K (1983) Scheduling subject to resource constraints: classification and complexity. Discrete Appl Math 5(1):11–24

Brucker P, Drexl A, Möhring R, Neumann K, Pesch E (1999) Resource-constrained project scheduling: notation, classification, models, and methods. Eur J Oper Res 112(1):3–41

Fliedner T (2011) Development and experimental investigation of scheduling approaches for the stochastic resource-constrained multi-project scheduling problem. Master Thesis, TUM School of Management, Technische Universität München, München
Graham RL (1966) Bounds on multiprocessor timing anomalies. Bell Syst Tech J 45:1563–1581
Hartmann S (1998) A competitive genetic algorithm for resource-constrained project scheduling. Naval Res Logist 45(7):733–750
Hartmann S (2002) A self-adapting genetic algorithm for project scheduling under resource constraints. Naval Res Logist 49(5):433–448
Hartmann S, Briskorn D (2010) A survey of variants and extensions of the resource-constrained project scheduling problem. Eur J Oper Res 207(1):1–14
Heller U (1981) On the shortest overall duration in stochastic project networks. Methods Oper Res 42:85–104
Herroelen W, Leus R (2005) Project scheduling under uncertainty: survey and research potentials. Eur J Oper Res 165(2):289–306
Jarboui B, Eddaly M, Siarry P (2009) An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems. Comput Oper Res 36(9):2638–2646
Kolisch R (1995) Project scheduling under resource constraints: efficient heuristics for several problem classes. Springer, Heidelberg
Kolisch R (1996) Serial and parallel resource-constrained project scheduling methods revisited: theory and computation. Eur J Oper Res 90(2):320–333
Kolisch R, Hartmann S (1999) Heuristic Algorithms for Solving the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis. In: Weglarz J (ed) Project scheduling: recent models, algorithms, and applications. Kluwer Academic Publishers, Norwell, pp 147–178
Kolisch R, Hartmann S (2006) Experimental investigation of heuristics for resource-constrained project scheduling: an update. Eur J Oper Res 174(1):23–37
Kolisch R, Sprecher A (1996) PSPLIB-A project scheduling problem library. Eur J Oper Res 96:205–216
Larranaga P, Lozano JA (2002) Estimation of distribution algorithms: a new tool for evolutionary computation. Kluwer Academic Publishers, Norwell
Möhring R, Radermacher F, Weiss G (1984) Stochastic scheduling problems I—general strategies. Math Methods Oper Res 28(7):193–260
Montgomery D (2009) Design and analysis of experiments, 7th edn. Wiley, Hoboken
Neumann K, Schwindt C, Zimmermann J (2003) Project scheduling with time windows and scarce resources, 2nd edn. Springer, Berlin
Pan Q-K, Ruiz R (2012) An estimation of distribution algorithm for lot-streaming flow shop problems with setup times. Omega 40(2):166–180
Rubinstein RY, Kroese DP (2004) The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning. Springer, Berlin
Sprecher A (2000) Scheduling resource-constrained projects competitively at modest memory requirements. Manage Sci 46(5):710–723
Stork F (2000) Branch-and-bound algorithms for stochastic resource-constrained project scheduling. Research report, 702, Technische Universität Berlin, Fachbereich Mathematik
Stork F (2001) Stochastic resource-constrained project scheduling. Doctoral Thesis, Technische Universität Berlin
Tsai YW, Gemmill D (1998) Using tabu search to schedule activities of stochastic resource-constrained projects. Eur J Oper Res 111(1):129–141
Wang L, Fang C (2012) An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem. Comput Oper Res 39(2):449–460
Zhang Y, Li Z (2011) Estimation of distribution algorithm for permutation flow shops with total flowtime minimization. Comput Ind Eng 60(4):706–718

Chen Fang received his PhD in Control Theory and Control Engineering from Tsinghua University. His research interest is in project scheduling based on computational intelligence.

Rainer Kolisch is Professor of Operations Management in the TUM School of Management at Technische Universität München in Germany. He obtained his Diploma degree in Industrial Engineering from Technische Universität Darmstadt and his Doctoral degree from Christian-Albrechts Universität Kiel. His current research is in service operations and project management.

Ling Wang is Professor in the Department of Automation at Tsinghua University. He received his PhD in Control Theory and Control Engineering from Tsinghua University. His research interest is in optimization and scheduling based on computational intelligence. He has authored over 260 papers and 5 books in this research field.

Chundi Mu is Professor in the Department of Automation at Tsinghua University. She received his BS in Automation from Tsinghua University. Her research interest is in control and optimization for engineering systems.